**HUMAN VS AI CHESS ENGINE**

**PROJECT PROPOSAL**

**Group Members:**

1. 221414 - Kanwar Junaid Islam
2. 221357 - Muhammad Haris Akhtar
3. 221390Jasim Ibrahim Malik

**Submitted to:**

Ms. Warda Aslam

****

# 1. Project Overview

This project proposes the development of a **smart chess engine** that combines **Constraint Satisfaction Problem (CSP) modeling**, **Alpha-Beta Pruning**, and **Iterative Deepening Minimax** to simulate intelligent AI gameplay against a human opponent. The application includes a **Windows Forms-based GUI frontend** in C# for user interaction and a **Python-based backend** that handles the game state and AI logic. The system is designed to intelligently prune illegal and non-optimal decisions using constraints and evaluate the best move

# 2. Motivation

Chess AI typically focuses on brute-force evaluation or deep search trees. While effective, they often lack explainable logic related to constraints like check, pinned pieces, or forced moves. This project aims to enhance the AI’s **understanding of legal play using CSP**, apply **Alpha-Beta Pruning** for decision-making efficiency, and integrate **Iterative Deepening Minimax** to balance depth and response time — making the AI **strong, fast, and logical**.

# 3. Objectives

* Design a **CSP module** to model and enforce core chess constraints.
* Implement **Alpha-Beta Pruning** to improve move selection efficiency.
* Add **Iterative Deepening Minimax** to allow time-bound progressive decision-making.

# 4. Scope

* Full support for single-player Human vs AI chess gameplay.
* Enforcement of core chess rules using CSP modeling.
* Decision-making via Alpha-Beta Pruning and Iterative Deepening.
* GUI to visualize the game board and handle user moves.
* No multiplayer or online play (can be added in future versions).

# 5. Technologies and Tools

|  |  |
| --- | --- |
| **Component** | **Technology** |
| Backend AI | Python 3, python-chess |
| CSP Modeling | Custom Python logic |
| AI Algorithms | Alpha-Beta, Iterative Deepening |
| Frontend GUI | Windows Forms (C#) |
| Communication | Flask REST API |
| IDEs | Jupyter Notebook, Visual Studio |
| Version Control | Git/GitHub |

# 6. Methodology

1. **CSP Modeling**
   * Variables: Chess pieces
   * Domains: Valid moves
   * Constraints: Avoid leaving king in check, obey pin logic, forced capture (optional)
2. **Alpha-Beta Pruning**
   * Traditional Minimax enhanced with pruning to reduce search space
   * Works on CSP-filtered legal moves
3. **Iterative Deepening Minimax**
   * Performs Minimax to increasing depths
   * Returns best move found within a time limit
   * Ensures responsiveness and deeper insight as time allows
4. **Frontend Development**
   * Built using Windows Forms
   * Sends human move to backend via API
   * Renders AI response on the board
5. **Integration & Testing**
   * Connect Python backend to C# frontend
   * Ensure real-time, valid gameplay
   * Test with various game states and edge cases

# 8. Expected Outcome

* Fully functioning **Constraint-Aware Chess Engine** with:  
  + Rule-enforced move legality via CSP
  + Efficient AI move generation using Alpha-Beta pruning
  + Real-time move decision via Iterative Deepening Minimax
  + Visual game interface built in Windows Forms
* Comparative insights between Alpha-Beta vs Iterative Deepening strategies

# 10. References

* Python-Chess Library: [https://python-chess.readthedocs.io](https://python-chess.readthedocs.io/)
* AIMA: Artificial Intelligence – A Modern Approach (Russell & Norvig)
* Alpha-Beta Pruning Algorithm Tutorials
* Microsoft Docs: Windows Forms
* Flask REST API Guide